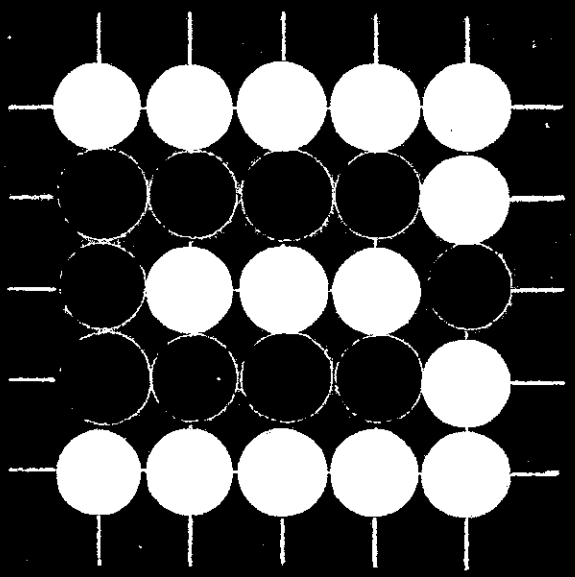
közlemények

15/1975



Felelős kiadó:

Dr. Vámos Tibor
igazgató

OPERÁCIÓS RENDSZEREK ELMÉLETE

Téli Iskola, Visegrád

MTA Számítástechnikai és Automatizálási Kutató Intézet
MTA Számítástudományi Bizottsága

Technikai szerkesztő:

Solt Jánosné
MTA Számítástechnikai és Automatizálási Kutató Intézete

757819 MTA KESZ Sokszorosító. F. v.: Szabó Gyula

TARTALOMJEGYZÉK

Operációs rendszerek (bevezetés) 7

Tomkó József:
Tömegkiszolgáláselméleti modellek számítógéprendszerek matematikai leírásában 9

Arató Mátyás:
Operációs rendszerek működésének diffúziós közelitése I.–II. 21

I.V. Szergienko, I.N. Paraszjuk:
Automatikus adatfeldolgozó rendszerek kialakitásának tapasztalatai 43

A. Wolisch:
Real–time operációs rendszerek valószinűségszámitási modelljei..... 55

K.Tantsher:
A CDC NOS – operációs rendszerek scheduling algoritmusai 73

Quittner Pál:
Patthelyzet operációs rendszerekben (Összefoglaló) 97

Tóth Beatrix – Tőke Pál:
A scratch–pool kihasználtságának mérése 103

Gyires Tibor:
A virtuális memória egy algebrai és egy valószinűségelméleti modellje 111

Benczúr András:
Nagy rendszerek software problémái 121

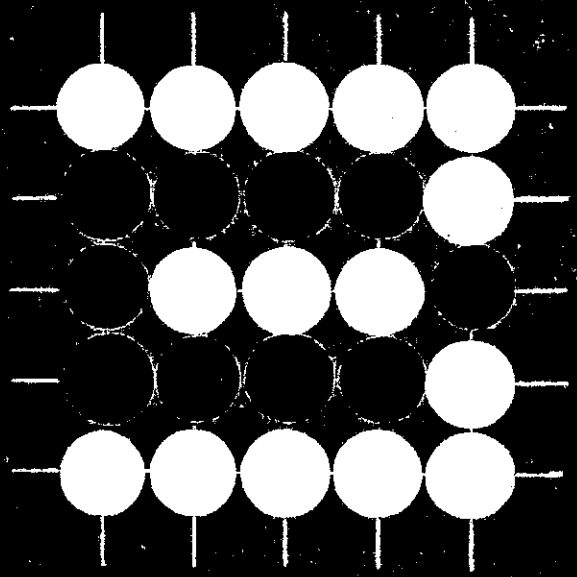
Knuth Előd:
A kölcsönös kizárási probléma elemei operációkkal történő megoldásairól ... 133

Knuth Előd:
Központi egység kihasználtságának szimulációs vizsgálata másodrendű autóregressziós folyamatok alkalmazásával 141

Szemelvények a kerekasztal megbeszéléseken elhangzott gondolatokból 147

közlemények

15/1975



MTA Számitástechnikai és Automatizálási Kutató Intézete, Közlemények 15/1975.

OPERÁCIÓS RENDSZEREK MÜKÖDÉSÉNEK DIFFUZIÓS KÖZELITÉSE I.

(Általános megjegyzések)

Arató Mátyás

MTA Számitástechnikai és Automatizálási Kutató Intézete

1. Tapasztalatainkról

Intézetünk Valószinüségszámitási és Matematikai Statisztikai Osztálya, valamint a Rendszertechnikai és Programozáselméleti (Software) Osztálya munkatársainak egy kisebb csoportja 1972-ben kezdett el foglalkozni operációs rendszerek kérdéseivel, valamint a megfelelő matematikai modellek tanulmányozásával és különböző tipusu feledatok megoldásával. Erre a következő okok miatt került sor: elsősorban a CDC – 3300-as gép operációs rendszerének ismeretében és a müködési tapasztalatok alapján olyan gyakorlati kérdések merültek fel (a scratch pool (SCR) dinamikus kezelése, a valódi multiprogramozás biztositása, a kvantum idő beállitása stb.), melyek nemcsak az operációs rendszer ismeretét, hanem mély elméleti meggondolások is igényelték. Másodsorban figyelembe kellett venni, ki kellett értékelni azokat a mérési eredményeket, statisztikákat, amelyeket a gép müködésének kezdetétől gyüjtöttünk. Végül a nemzetközi folyóiratokban, konferenciák kiadványaiban megjelent cikkek, az intézetben elhangzott előadások tömegesen vetettek fel olyan kérdéseket és ismertettek olyan megoldásokat, amelyek közel álltak az általunk is tapasztalt tényekhez, problémákhoz.

Mindezek alapján célul tüztük ki az oprációs rendszerek egyes részfolyamati matematikai leirásának megadását. A számitások alapján konkrét, a felhasználók számára haszonnal is járó célkitüzések megoldását is feladatainknak tekintettük. Az utóbbi feladatra két lehetőség is mutatkozott:

1. Az output (elsősorban nyomtatás) disk igényeinek dinamikus kezelése.
2. A gép bővitése alkalmával a paraméterek olyan beállitása, hogy valódi multiprogramozás valósuljon meg. Ez a gép központi (CPU, Central Processor Unit) egységének mintegy 20-30%-osan jobb kihasználását tette lehetővé, előzetes számitásaink szerint.

Az 1. pont alatti feladat részbeni megoldását adja a MASTER 4.0 operációs rendszer. Az általunk (lásd Tőke [20])* kidolgozott elvek alapján müködő változat még nem készült el. A 2. pont alatti célkitüzéseket olyan intézkedésekkel valósitottuk meg, melyek a felhasználók számára nem jelentettek a gépbővités során megszorításokat, de a kényelmük sem növekedett nagy mértékben. A valódi multiprogramozást (átlagosan 2–3 programmal) sikerült elérni.

*Az irodalmi hivatkozásokat a cikk második (angol nyelvü) része végén egységesen adjuk meg.

A CDC gép felhasználói programjai rövid jellemzését az alábbiakban lehet megadni:

A kialakult rendszer szerint természetes követelmény annak megvalósitása, hogy a program–próbák (debugging) ideje alatt lehetőleg nagy periféria–igényü programok is fussanak. Erre az ad lehetőséget, hogy a compute–idő mintegy másfélszerese a csatorna–időnek, igy elsősorban a compute–idő kihasználásával kell törődni. Nagy periféria–igényü, hosszu feldolgozási–igényü programokkal a gép féléves müködése óta rendelkezünk.

A SCR jelenlegi felhasználása két egymástól teljesen különböző felhasználási elv alapján történik. A számolásos (elsősorban FORTRAN-ban iródott) jellegü feladatoknál a SCR felhasználás véletlenszerü és véletlenszerü felszabaditásokkal jár. Az összigény nem jelentős, igy az ingadozás átlaggal és szórással jellemezhető volt. A nagy rendezői és kiirási munkák esetén a SCR felhasználása más-más módon valósul meg. Kiiratásoknál a kiirandó adatok véletlenszerü (feltehetően független exponenciális eloszlások szerint) jelennek meg. Ennek az igénynek jó kielégitése a szeletelt nyomtatás segitségével oldható meg. Rendezési (szortolási) feladatok megoldásánál a SCR igény egy nagyterjedelmü igény formájában, véletlen időtartamra jelenik meg és ezután a feladat további részében ez a terület nem kerül felhasználásra. E két nagy, különböző tipusu, SCR igény statisztikai leirása a dinamikus SCR felhasználás bevezetését teszi szükségessé. Számitásaink és méréseink szerint a SCR felhasználását a következő szabályokkal lehetett korlátozni. Az egy programban felhasználható SCR terület nagysága nem haladhatja meg a 140 szegmens értéket. Az output igényt esetleg külön igény formájában lehet megadni jelezve az output igény nagyságát. A dinamikus SCR felhasználása esetén a programok rendelkezésre álló SCR területét 250-300 szegmens körül lehetett meghatározni. Ez biztositotta a különböző tipusu programok egymás meletti futhatóságának feltételét.

A fenti adatokat folyamatos statisztikai felmérések végzésével egy kiértékelő program alapján kaptuk (lásd Tóth B. – Tőke P. [22]). Ezeknek a statisztikáknak az alapján megállapitható, hogy a gépi kihasználás legfontosabb paraméterei lényegében a várt értékek körül mozogtak és lehetőség volt a kiértékelések alapján extrapolációt végezni a gép leghasznosabb kihasználására vonatkozóan a bővités után.

A gyakorlatban elsősorban a programok száma szerinti statisztikákkal dolgoztunk és nem vettük figyelembe (mivel erre nem volt lehetőségünk) az egyes programfajták milyen hosszu ideig foglalják le a gépet. Erre csak a következő durva statisztikát tudjuk adni. A gép idejének mintegy 1/3 részében szervezési feladatok, tehát sok nyomtatást igénylő feladatok futnak, további 1/3 részban számitás–igényes feladatok futnak, mig a megmaradó 1/3 gépidőt kisebb programok töltik ki. A legutóbbi csoport esetén van értelme a programok száma szerinti eloszlásokkal is foglalkozni.

A gép müködésének matematikai leirásához szervesen hozzátartozik a job–folyam (vagy program–folyam) matematikai leirása is.

A job – folyam statisztikai viselkedésének ismeretében olymódon kell a jobok keverését elvégezni, hogy az összes job együttes átfutási ideje minimális legyen, azon kiegészitő feltétel mellett, hogy egy hét alatt minden beérkező job lefusson. A heti felosztáson, keverésen kivül

Jobok napi keverését (prioritásuk adása, napi legalább egyszeri futás biztositása stb.), az operátorok, illetve az erre kijelölt vezető végzi. Ezen kivül szükség van rövidebb időszakon belül (pl. óránként) algoritmikus job keverésre is.

A job–folyamot egy többdimenziós időben változó véletlen igénybeérkezésnek tekintjük, melynek kiszolgálása, a kiszolgálási idő kezdete, a szervezéstől, keverési eljárásoktól és a gépben működő operációs rendszertől függ. A job–folyamot ilyen sztochasztikus folyamatnak tekintve, és figyelembe véve, hogy a kiszolgálás folyamán a gép prioritási rendszerétől függően változik az átfutási időtartam, helyesnek látszott egy sztochasztikus modell alkalmazása. Mindezek annak ellenére is jó közelitésként használható, hogy a gép belső müködése szigorú logikai felépités alapján történik.

A job–folyam és a belső müködés egy sztochasztikus modellje megtalálható Arató – Knuth – Tőke [3] és Knuth [15], [16] cikkeiben, igy erre részletesen nem térünk ki.

2. A matematikai modellekről

Az alábbiakban a számitógép, illetve annak részei müködésének egyes matematikai modelljeivel foglalkozunk.

a.) A matematikai leirások közül első helyen kell kiemelni a tömegkiszolgálási modellek alkalmazását. Ennek összefoglalása a majdnem teljes irodalom megtalálható Kleinrock-nak az 1974 évi IFIP kongresszuson elhangzott előadásában [12]. Az intézetben végzett ilyen irányu vizsgálatokról Tomkó [21] cikke ad felvilágositást.

b.) A "telitett" tömegkiszolgálási prolémák vezettek el analitikus uton a diffuziós közelitések alkalmazásához. Ez a leirás található meg Gaver – Shedler [8],[9] valamint Kobayashi [13], [14] cikkeiben.

c.) A számitógépben lejátszódó folyamatok sztochasztikus közelitése az időintervallum megfelelő megválasztása esetén lehetővé teszi a közvetlen diffuziós leirások használatát. Erre tettünk javaslatot korábbi cikkeinkben: Arató [1], [2], Arató – Knuth – Tőke [3]. A gép belső müködését leiró szochasztikus folyamatok paraméterei ez esetben a job-ok függvényei azaz ismét véletlen paraméterek. A belső prioritásos rendszer kialakitása, a kvantum megválasztása a statisztikák állandó megfigyelésével valósitandó meg.

A matematikai, különösen a stasztikai jellegü, modellek nagy része jelenleg a sorbanállás matematikai elméletének alapján áll. Ugyanúgy mint más elméletek esetén is ezek a modellek sohasem felelnek meg teljes pontossággal a valódi rendszereknek, hanem azok lényegét kivánják megmutatni. A legfontosabb feltételeket és következtetéseket a meglévő adatok sokaságán nem szokás ellenőrizni. Ez igen nehézkes statisztikai vizsgálatok bevezetését jelentené. Ebben a cikkben (mindkét részében) a multiprogramozású számitógépek belső müködése leirására egy diffuziós tipusú közelitő modellt ismertetünk. Egy ilyen modellből származó következtetések

könnyebben ellenőrizhetőek, másrészről ezen a közelitési alapon nyílik lehetőség a sztochasztikus szabályozás, nem lineáris filtráció és más eredmények alkalmazására. Jólismert, hogy a sorbanállás matematikai elméletének eredményei nem egyszerüek és éppen ez gátolja gyors felhasználásukat is.

Korábbi dolgozatainkban (lásd pl. [1], [2], [3], [15], [16]) megmutattuk, hogy a multiprogramozásu számitógépek érdekes jellemzői adhatók meg néhány közvetlen diffuziós közelités analizise utján. Ezek az erdmények felhasználhatók a CPU (a központi egység, central processor unit) vagy a DTU (ádatátviteli egység(ek), data transmission unit) müködésének értékelésében.

Az általunk javasolt sztochasztikus modell, mely az operációs rendszer müködését kivánja közelitően megadni, lehetőséget nyújt a számitógép hosszabb időtartamú müködtetése esetén (nagy megterhelésü job–folyamot feltételezve) a müködés általános értékelésére és szabályozására, néhány paraméter segitségével.

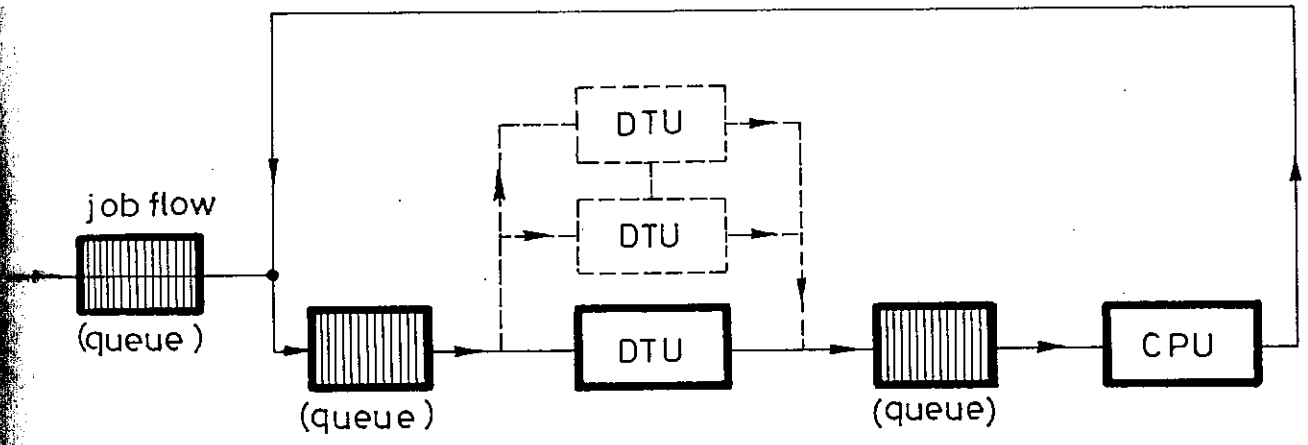
Ebben az esetben a statisztikai analizis során lehetőség van a felesleges matematikai nehézségek elkerülésére, melyek a ciklikus sorbanállási modelleknél jellemzőek (lásd pl. Gaver – Shedler [8], [9] vagy Reiser – Kobayashi [17]). A ciklikus sorbanállási modelleknél a nehézségek legjelentősebb forrása az adatátviteli müveletek nem–Markov mivoltából adódik.

A ciklikus sorbanállási modellek diffúziós közelitése is mutatja ezen hozzáállás egyszerü következtetési eredményeit (lásd [8], [13]). Ezek az eredmények természetes módon vetik fel annak szükségességét, hogy az operációs rendszerek müködésének leirására közvetlen diffúziós közelitéseket használjunk (a sorbanállási modellek nélkül). Ebben a cikkben egyszerü bizonyitásokat kivánunk bemutatni a diffuziós közelités jóságára és azt is megmutatjuk, hogy modelljeink egyszerüen kezelhetőek multiprogramozású rendszereknél. A statisztikai kiértékelések és a nemlineáris sztochasztikus szabályozás lehetőségeinek bemutatása ugyancsak feladata a cikknek. Standard ismereteket tételezzünk fel és a szokásos leirást alkalmazzuk (a) a programok viselkedésére a központi egységben, (b) az információ elérésére az adatátvitelnél, (c) a tárolási hierarchiák megadására. Eredményeink hasznosaknak tünnek egyes elemek vagy nagyobb egységek müködési paraméterei kiválasztására is.

A [15], [16] dolgozatainkban javasolt valószinüségi leirást használjuk a továbbiakban. Feltételezzük, hogy nemcsak a Brown–mozgás, hanem általános diffuziós folyamatok is felléphetnek a számitógépek müködési leirásában. Ennek egyik bizonyitéka, hogy gyakran mérhetőek a kovariancia függvények és ezek lényeges eltérést mutatnak a Brown–mozgás kovariancia függvényétől.

A modellben szereplő programok (job-ok) a központi memória valamely részében vannak elhelyezve, éppúgy mint az operációs rendszer programjai. A lehetséges elhelyezhető programok száma a multiprogramozás foka, amelyről feltesszük, hogy konstans és K-val jelöljük. Ez természetes feltevés leterhelt rendszereknél.

Feltételezzük, hogy (mint pl. [9], [13]-ban) a K program a központi egység és az adatátviteli egységek által alkotott ciklusban müködik. Az egyes programok vagy a központi egységre várnak vagy ott állnak kiszolgálás alatt, melynek végeztével adatátviteli egységre kerülnek. Az adatátviteli egységről, a szükséges információ megkapása után, a program visszatér a központi egységre. Ez a ciklikus folyamat meghatározatlan ideig folytatódik. Ha egy program kiszolgálása befejeződik és kikerül a rendszerből, helyére a program–folyamból új program kerül a központi memóriába. Az 1. ábra szemléletesen mutatja a folyamatot.



1. ábra

A programok viselkedésére vonatkozóan a következő feltevésekkel élünk:

a.) a központi egységben történő kiszolgálási idők független valószinüségi változók (esetleg nem azonos eloszlásuak);

b.) az adatátviteli egységek kiszolgálási idejei ugyancsak független valószinüségi változók;

c.) a központi egység és adatátviteli egységek kiszolgálási idejei egymástól teljesen függetlenek.

Megjegyezzük, hogy függő valószinüségi változók esetén nem jutunk Brown–mozgás közelitére, hanem általánosabb diffúziós folyamatra. Feltételezzük, hogy a központi egység előtti sor és az adatátviteli egységek előtti esetleges sor kiszolgálása a FIFO (először érkezett–először kerül kiszolgálásra) elv alapján történik, ha ezt másként nem szabályozzuk.

Az előbbiekben leirtak szemléltetésére és a nagyságrendek érzékeltetésére példaként vázlatosan ismertetjük az akadémiai CDC gépnek a leiráshoz szükséges adatait.

A CDC – 3300 gép egy négy programot ($K = 4$) megengedő rendszerének sematikus általánositásából adódó modell a következő. A kvantum idő, melyet Q-val jelölünk jelenleg 1/10 sec. Az I/O megszakitások adminisztrációja, mind az input, mind az output adminisztrációnál $\frac{1}{2}$ μ sec. Ezen adminisztráció végzése alatt nincs még kvantum megszakitás sem. Esetünkben az adatátviteli egységeket a csatornák jelentik, melyek a perifériális egységekkel kötik össze a központi egységet. Feltételezve egy abszolut prioritásos kiszolgálási rendszert (mely

I/O megszakitásnál érvényes) az egyes programokról a következőket tételezzük fel:

az 1. program (job) CPU kiszolgálási ideje 1μ sec átlagos idejü és a csatornán való tartózkodás ugyancsak 1μ sec átlagos idejü valószinüségi változó.

a 2. job 5μ sec átlagos CPU igénnyel és 4μ sec átlagos idejü csatorna tartozkodásu:

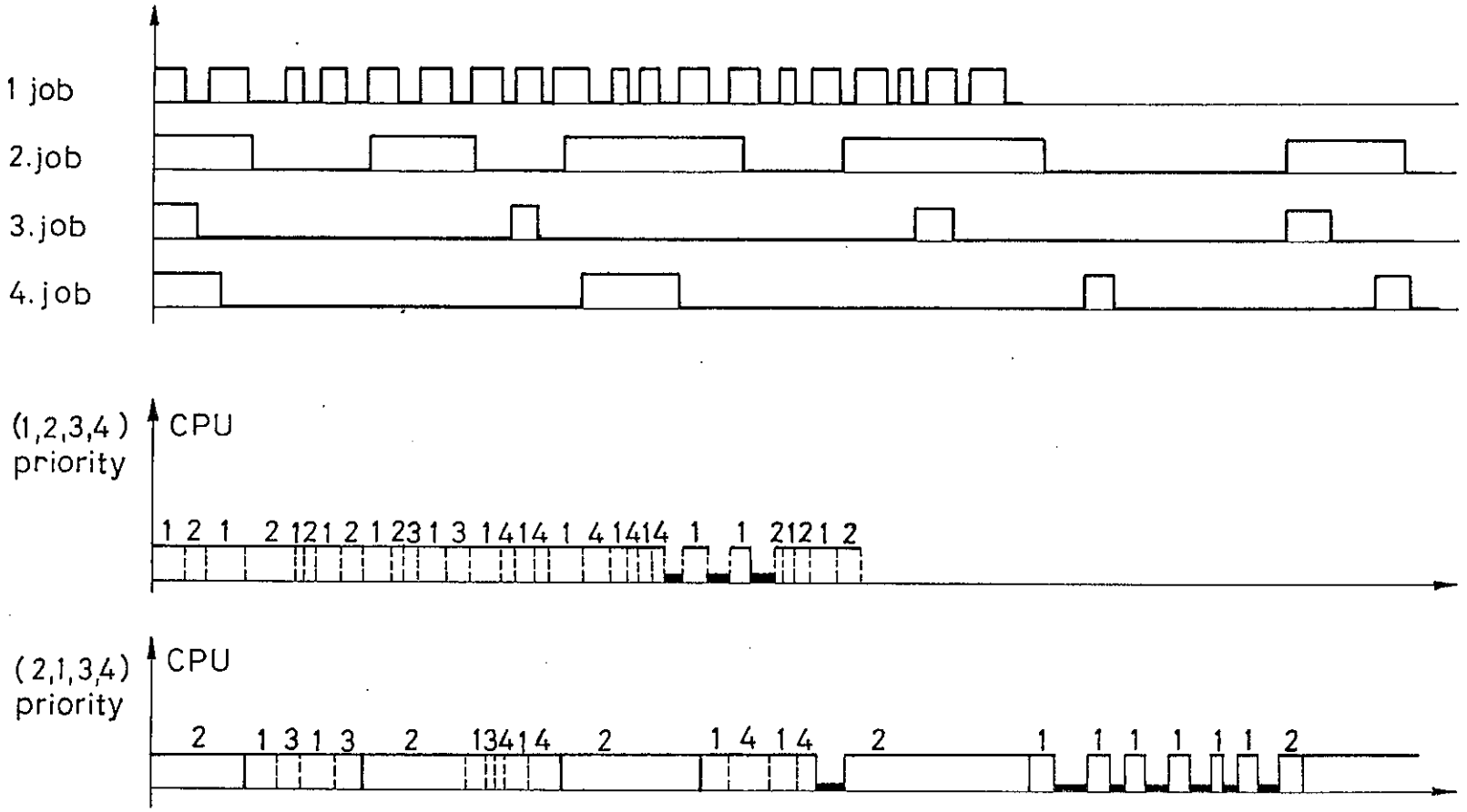
a 3. job 11μ sec átlagos idejü CPU kiszolgálással míg a csatornán 10μ sec átlagos idővel van,

a 4. job 2μ sec átlagos idővel a CPU-ban és 10μ sec átlagos idővel a csatornán van.

Ha feltételezzük, hogy az abszolut prioritási rendszer az első esetben (1, 2, 3, 4) a második esetben pedig (2, 1, 3, 4) és a CPU kihasználtságot és a csatornák foglaltságát a két prioritás esetén valamint a round–robin elv alapján egy kvantum megszakitás környezetében ábrázoljuk, érdekes következtetésekre jutunk (v.ö. 2. ábra).

A rövid csatorna idejú és sok megszakitásos programok dominálnak. A szemlélet alapján azt lehet várni, hogy a CPU kihasználtsága annál a prioritási rendszernél jobb, melyben az 1. job megelőzi a 2-t. Ekkor ugyanis a 2. job csatorna ideje, az 1. job csatorna ideje és CPU ideje alatt is folyik. Ellenkező prioritás esetén az 1. job csatorna ideje lényegében a 2. job csatorna ideje alatt folyik és nem folyik a CPU ideje alatt. A gyors I/O változásnál a kvantumnak nincsen jelentős szerepe a CPU kihasználtságában. A kvantum a jelentős CPU igényü feladatoknál játszik lényeges szerepet, amikor az I/O megszakitások és CPU idők nagyságrendje a kvantum, vagy annál nagyobb. Ilyen esetekben a kvantum jelentősen módositja a sorrendet, és a prioritás elveszti elsődleges meghatározó szerepét. A 2. ábrán egy szimulációs eredményt ábrázoltunk, ahol az eloszlások exponenciálisak.

A rendszer leírásának első közelítésében nem vesszük figyelembe az I/O megszakitásokkal járó adminisztrációt, mivel a jelenlegi rendszerben azt a CPU végzi és a CPU idejében is történik az elszámolás. Amennyiben a megszakitások száma nagy, és ez az adminisztráció jelentős, természetes az olyan rendszerek kiépitése, ahol ezt az adminisztrációs feladatot nem a CPU, hanem egy külön egység végzi.


1 job
2.job
3.job
4.job
(1,2,3,4)
priority
CPU
1 2 1 2 12 1 2 1 23 1 3 1 4 14 1 4 14 14 1 1 2 12 1 2
(2,1,3,4)
priority
CPU
2 1 3 1 3 2 1 3 4 1 4 2 1 4 1 4 2 1 1 1 1 1 1 2

2. ábra